submits code, but also for ensuring the overall quality of the software.

The importance of community was supported in a study of contributors to the Linux operating-system kernel project. Guido Hertel, Sven Niedner and Stefanie Herrmann found that the more contributors identified themselves as "Linux developers," the higher their level of involvement and the greater their efforts to code for the project. The Linux project and communities surrounding it have the advantage of being well known, and the project's positive image can create a sense of responsibility and loyalty among developers. They also found contributors to be motivated by group-related factors such as their perceived indispensability to the team. Lastly, they found that contributors had important pragmatic motives to improve their own software. This is key to understanding why many skilled software developers work for free. Nic Franke and Eric von Hippel studied the security needs of those who use Apache Web-server security software. They found that because Apache is open-source software, users can modify it to better fit their individual needs and that users who do so are more satisfied.

While these motives explain, to a great degree, the benefits and incentives of becoming involved in open-source projects, the issue of how developer communities protect the fruits of their labor is a subject of some concern. Unlike in commercial development — where intellectual property law protects the rights of authors to appropriate economic returns from their innovations — open-source licenses are designed to guarantee the rights of *future users* against appropriation. Siobhan O'Mahony points out that open-source software contributors have an active concern that their work remains part of the commons, and they zealously protect their work to this end. Open-source project members encourage compliance with the terms of project licenses in various ways. They may exercise sanctioning via online discussions and may use brands and logos to ensure that the intellectual property they have contributed remains in the commons.

OPEN-SOURCE LICENSES ARE DESIGNED TO GUARANTEE THE RIGHTS OF *FUTURE USERS* AGAINST APPROPRIATION.

The Innovation Process

Open-source projects can be started by anyone with the appropriate programming skills and motives. Typically, an entrepreneur with some workable code or an idea for a software project launches a message on one of the many Web-based collaborative communities, such as freshmeat.net or Geocrawler.com. If others find the code useful or the idea interesting, they join in by contributing code, fixing bugs or other problems in the software, providing comments, ideas, references to other projects and so on. Over time, the number of contributors can grow substantially, and the project may use one of the many technical infrastructures available on the Web to host and monitor changes to the emerging software. The entrepreneurs also normally set up a project mailing list for posting questions and answers or messages pertinent to developing the software.

The first challenge for the open-source entrepreneur is mobilizing top-notch programmers. In some cases, when no viable commercial alternatives exist or when they are too expensive, it may, in fact, be easier to attract top programmers to open-source projects than to commercial development ventures. Developers of standard off-the-shelf software often find it difficult to judge whether a product feature will have a major impact on satisfying user needs, and they also recognize that users often have difficulty expressing their needs. Eric von Hippel has noted that such "sticky" information is costly to retrieve because understanding a user's problems requires that the manufacturer "dwell in the context of the user" for a prolonged period. As a result, the high development (and marketing) costs for standard packaged software typically cause software firms to spread these

costs among a large population of users. Companies will therefore seek information about "average user needs and problems," and this affects product development. In some cases, companies supplement this with help from external opinion leaders in order to ultimately strengthen the product design and identify bugs in early releases. For some types of software, this way of optimizing the innovation process leads to market failure and will spur the interest of developers in joining open-source software projects, in which they can code to meet their own practical and technical needs.

Another challenge for open-source entrepreneurs, as has been discussed by Michael Cusumano, is organizing the innovation process properly. For-profit programming companies often seek to reduce development costs and control quality by closely monitoring what programmers do and how they do it. To secure returns on investment in innovation, most companies try to seek out and recruit the most outstanding software talent, bind them by contract and take steps to minimize opportunism. (R.D. Austin has explored in detail the relationship between the software developer and the firm.) Additionally, software companies attempt to contain costs as well as prevent spillover of knowledge, technology and other secrets to competitors by encouraging specialization and division of labor among developers.

By contrast, the relationship between the open-source software project and its participants is largely voluntary and not regulated by formal contract. The initiator of a project might become well known in the public domain, such as Linus Torvalds who created the Linux operating-system kernel (the central module responsible for such functions as memory, process and disk management); but an initiator cannot legally force participants to continue or increase their efforts in the project. Recent work by Karim Lakhani and Eric von Hippel and by Jae Yun Moon and Lee Sproull shows that contributors to open-source software projects value a sense of ownership and control over the work

product — something they do *not* experience in programming work carried out for hire. For this reason (among others), participants of open-source software projects also do not take any particular action to minimize "free riding" (the downloading or "consumption" of remote files without reciprocal uploading or "production" of useful contributions). Project learning in the open-source world is captured in the chronology of e-mail exchanges and in the source code that is open for all to see; there is no need to contain or merge information according to a formal division of labor. By means of this transparency, a project accumulates the development efforts of volunteer users and seems to encourage the software's diffusion in order to build a developer's reputation and spread the products.

Whereas a firm may secure the best talent through the processes of professional recruiting, there is no formal recruiting in open-source software projects. This could lead to less talented individuals participating in open-source endeavors and eventually to compromises in the quality of the software. But research has shown that developer communities are meritocracies, in which technical knowledge and expertise determine a contributor's impact on the software design. Sebastian Spaeth, Karim Lakhani and I studied a sophisticated peer-to-peer software project named Freenet and found that only 30 people had the right to include code in the official version of the software. To become one of these core developers, participants had to demonstrate a considerably higher level of technical activity than other contributors. The open-source organization also self-allocates talent. As noted in our work, as well as in two other studies by Stefan Koch and Georg Schneider and by Bruce Kogut and Anca Metiu, people are not assigned tasks on the basis of predefined labor schemes, instructions and directives, but rather on the basis of interest and self-selection. This drives a high level of specialization, but implies that potentially useful software modules may never be developed.

THERE ARE GREAT ADVANTAGES TO A MODEL WHEREBY RESOURCES USED FOR INNOVATION ARE WIDELY DISTRIBUTED THROUGHOUT THE WORLD.

A View to the Future

What does the open-source software phenomenon imply for future business activities? In the short to medium term, some managers may encourage the use of open-source software in their own firms. Others may attempt to build a business based on distributing and servicing open-source software. U.S.-based Red Hat and German-based SuSE, which distribute Linux software, serve as templates for such activity. Other companies may sell computer hardware running open-source software, such as IBM, which offers Linux as an option. Managers may also try to reduce development costs and boost software standards by using the open-source software development model. Such an example is Sun Microsystems' decision to rely on open-source methods to develop and distribute Java.

The open-source software movement also provides important management lessons regarding the most effective ways to structure and implement innovation. There are potentially great advantages (and perhaps some disadvantages) to a model whereby resources used for innovation are widely distributed throughout the world. (See also Henry Chesbrough's article, "The Era of Open Innovation," p. 35.) The lessons of open-source projects demonstrate the value of specialization through self-selection and how norms of meritocracy and peer recognition help ensure product quality. To be sure, finding the right blend of incentives to encourage innovation is not an easy task. In the long run, however, managers may recognize that offering a mix of motives is the best way to encourage innovation; a mix that ranges from extrinsic and monetary-based incentives to the fulfillment of more-intrinsic needs, such as enhanced reputation among peers and community identification — that is, a sense of belonging.

Suggested Reading

Hackers: Heroes of the Computer Revolution, **Steven Levy** (New York: Anchor/Doubleday, 1984)

The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, **Eric Raymond** (Sebastopol, California: O'Reilly, 1999)

Rebel Code: Linux and the Open Source Revolution, **G. Moody** (Cambridge, Massachusetts: Perseus Publishing, 2001)

Networks of Innovation, **I. Tuomi** (Oxford University Press, 2002)

"The Open Source Definition," B. Perens (1998), http://perens.com/Articles/OSD.html

"Why Open Source Software Can Succeed," A. Bonacorsi and C. Rossi *Research Policy* 32 (2003, in press)

"How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies," J. West *Research Policy* 32 (2003, in press)

"Institutional Entrepreneurship in the Sponsorship of Common Technological Standards: The Case of SUN Microsystems and Java," R. Garud, S. Jain and A. Kumaraswamy *Academy of Management Journal* 45, no. 1 (2002): 196-214

Reprint 4432

MITSloan
Management Review

Reprints/Back Issues

Electronic copies of SMR articles can be purchased on our website:
www.mit-smr.com

To order bulk copies of SMR reprints, or to request a free copy of our reprint index, contact:

MIT Sloan
Management Review
Reprints
E60-100
77 Massachusetts Avenue
Cambridge MA 02139-4307
Telephone: 617-253-7170
Toll-free in US or
Canada: 877-727-7170
Fax: 617-258-9739
E-mail: smr@mit.edu

Copyright Permission

To request permission to copy articles, contact:

P. Fitzpatrick,
Permissions Manager
Telephone: 617-258-7485
Fax: 617-258-9739
E-mail: pfitzpat@mit.edu